# IST 718:
# Big Data Analytics

## UNIT 5-1 Statistical and Machine Learning

*Note: MATERIAL here is from a mix of Readings & sources as listed in References Slide(s)*

## Statistical and Machine Learning

- Statistical Learning vs. Machine Learning
- Regression vs Classification
- Bias / Variance Tradeoff
- Supervised vs. Unsupervised Learning
- Least Square Error Loss Function (LSE)
- Parametric vs Non-parametric Models
- Maximum Likelihood Estimation
- Linear Regression & Least Squares
- Measuring Generalization Performance
- Confusion Matrix
- The Bayesian classifier

## What is Learning?

- Merriam-Webster:
    1. The act or experience of one that learns
    2. Knowledge or skill acquired by instruction or study
    3. Modification of a behavioral tendency by experience (such as exposure to conditioning)

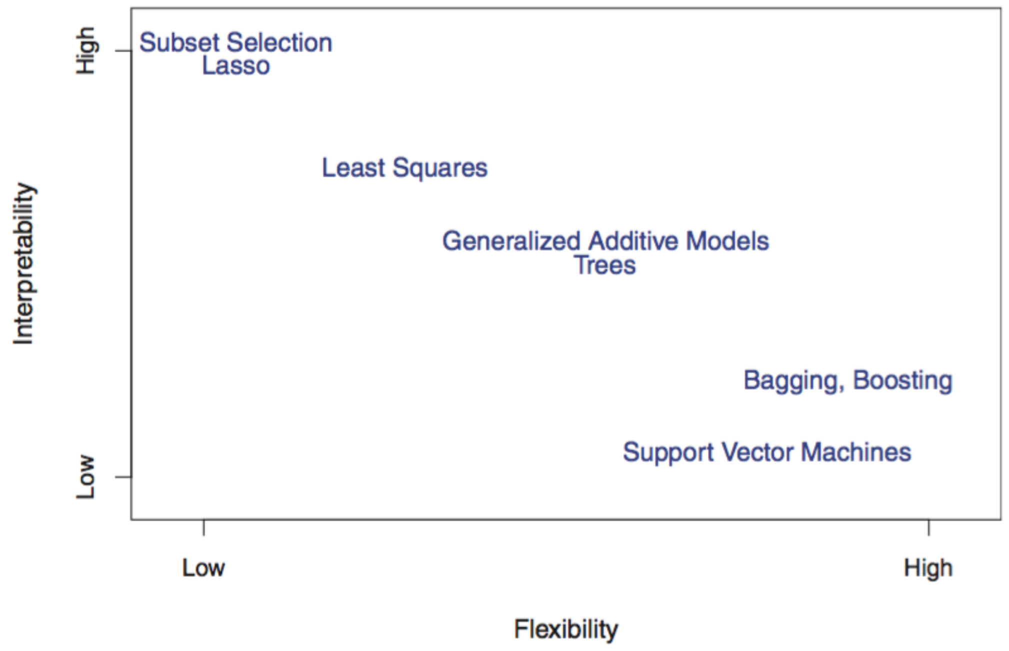
## Statistical Learning vs. Machine Learning

**From ISLR:**

- Machine learning arose as a subfield of Artificial Intelligence.
- Statistical learning arose as a subfield of Statistics.
- There is much overlap — both fields focus on supervised and unsupervised problems:
  - Machine learning has a greater emphasis on large scale applications and prediction accuracy.
  - Statistical learning emphasizes models and their interpretability, and precision and uncertainty.
- But the distinction has become more and more blurred, and there is a great deal of "cross-fertilization".
- Machine learning has the upper hand in Marketing!

## Accuracy vs. interpretability tradeoff

- In general:
  - Simple models are less accurate but more interpretable
  - Complex models are more accurate but less interpretable



## Machine Learning Perspective

- In his Machine Learning textbook Professor Mitchell states:
  - "The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience."
- http://www.cs.cmu.edu/~tom/mlbook/keyIdeas.pdf

Tom M. Mitchell
Carnegie Mellon University

## Machine Learning Restated

- Restated: How can a machine learning algorithm use **experience** to **improve future performance**
  - Experience = Data
  - Future = Data not yet seen
  - Performance = Error function or loss function

Tom M. Mitchell
Carnegie Mellon University

## Regression vs Classification

- When the variable we are trying to predict (*Y*) is quantitative (like a real number), then we talk about *regression*
- When the variable is categorical (like cat vs. dog), then we talk about *classification*
- Can you guess supervised / unsupervised and regression/classification?
  - User behavior clustering
  - Sentiment analysis
  - Credit score
  - Predicting whether an image contains a cat or not.

## Model Bias and Variance

- Variance: refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
- Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
- **High variance** means that small changes in the training data can result in large changes in the prediction.
- **High bias** is a high error introduced by approximating a complicated real-life problem with a simple model (like linear regression).

# Bias / Variance Tradeoff

- High bias: Straight orange line in the left hand plot.
- High variance: blue and green lines in left hand plot.
- Right hand plot: Gray line shows that as we continue training, training data MSE continues to decrease.
- Right hand plot: Red line shows that as we continue training, validation MSE decreases to a minimum and then starts increasing.
- Overfitting happens as the gray line decreases while the red line increases in the right hand plot.

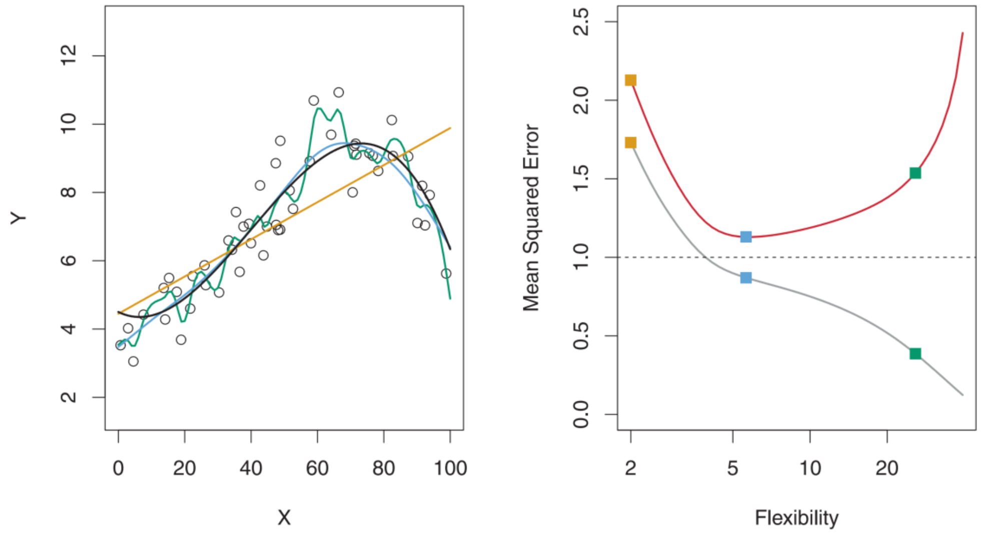


**FIGURE 2.9.** Left: *Data simulated from $f$, shown in black. Three estimates of $f$ are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

## The Bias-Variance decomposition

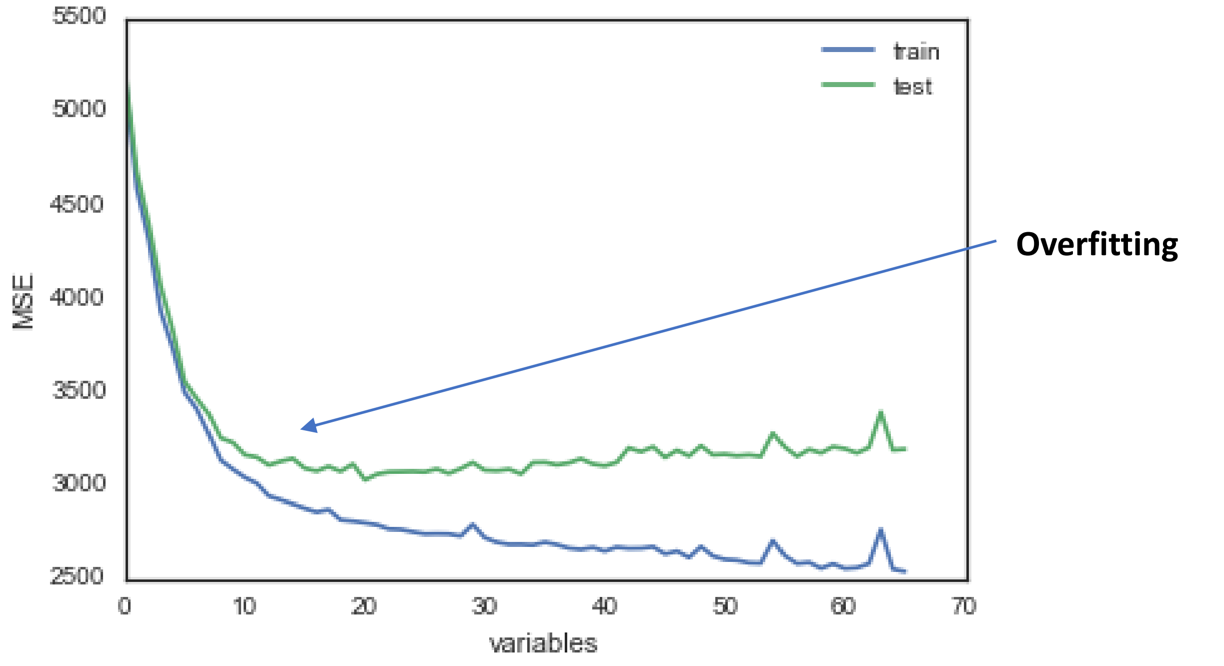
- In general, more complex models have low bias and high variance
- Vice versa, simple models have high bias and low variance
- This is a **fundamental tradeoff**

## The Bias-Variance decomposition (2)

- An example with the diabetes dataset
- By adding a polynomial expansion to the variables:
  - bmi, age, bmi*age, bmi^2, age^2, etc.
- We will have a new $X_{new}$ matrix with $p_{new} = 2p$ + p(p-1)/2 dimensions and run the following procedure:

1. Repeat many times
   1. Randomly split the data into training and testing
   2. For $k$=1 to $p_{new}$
      1. Fitting model with $k$ randomly selected features and predicting a random
      2. Estimate training and testing MSE

## The Bias-Variance decomposition (3)

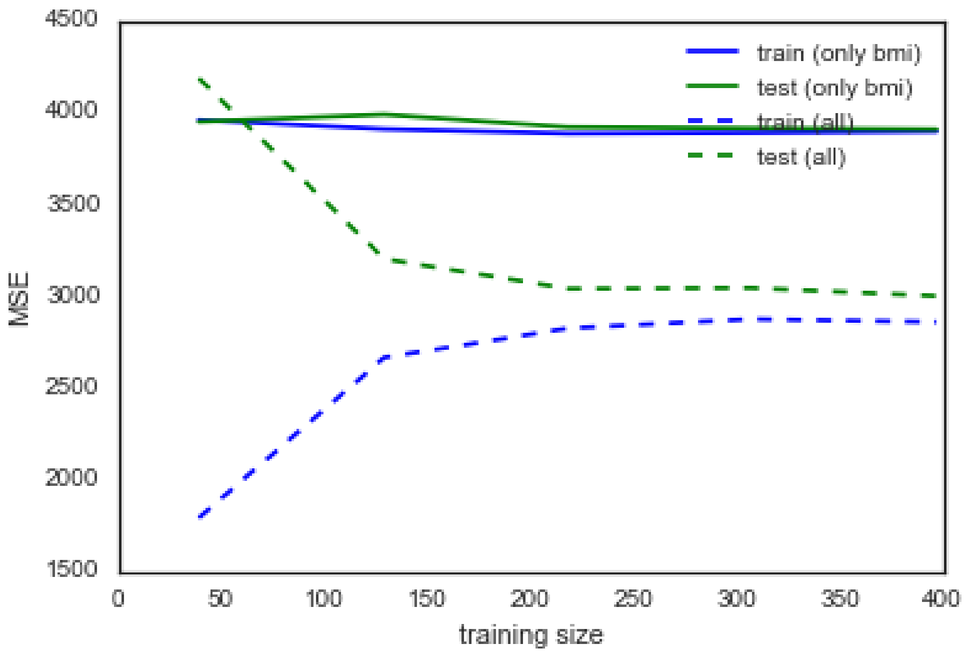


Low variance
High bias

High variance
Low bias

## Bias-Variance decomposition: the learning curve

- Simple model doesn’t learn much after 50 examples
- Complex model keeps learning
- Simple model has better performance with small datasets
- Complex model has better performance with big dataset



## The Bias-Variance Trade Off Equation

- The test MSE (red line in ISLR figure 2.9 in above slide) follows the following equation (ISLR equation 2.7):

$$Test_{MSE} = E[\left(y_0 - \hat{f}(x_0)\right)^2]$$
$$Test_{MSE} = \left(E\left[\hat{f}(x_0)\right] - f(x_0)\right)^2 + E\left[E\left[\hat{f}(x_0)\right] - \hat{f}(x_0)\right]^2 + Var(\epsilon)$$
$$Test_{MSE} = \mathrm{Bias}(\hat{f}(x_0))^2 + \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Var}(\text{Irreducible error})$$

- The notation $E[\left(y_0 - \hat{f}(x_0)\right)^2]$ defines the *expected test MSE*, and refers the the expected test MSE that we would obtain if we repeatedly estimated *f* using a large number of training sets, and tested each at $x_0$.

## Minimizing the Bias-Variance Trade Off Equation

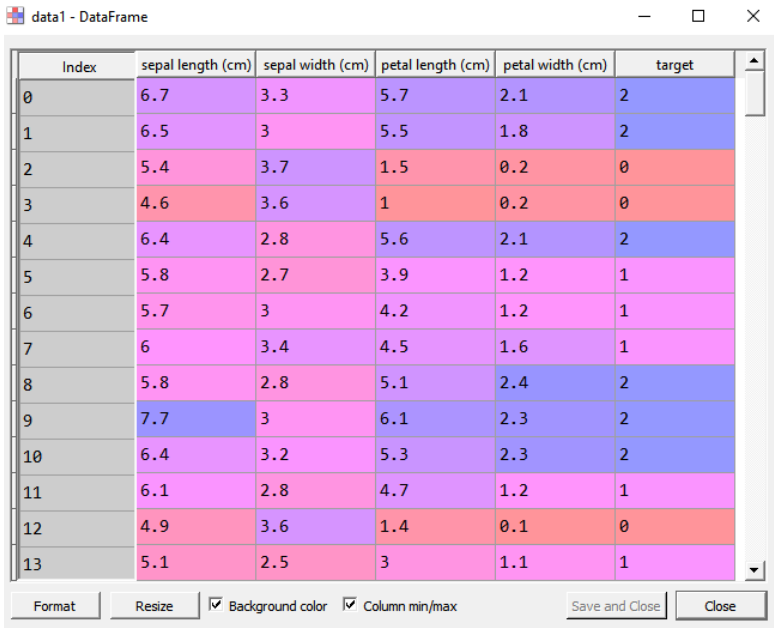
- In order to minimize test MSE, we need to select and tune a statistical learning method which simultaneously minimizes bias and variance.
- Variance and squared bias are non-negative so test MSE will never fall below the irreducible error term.

## Example Model Variance Characteristics

- In linear regression, variance is proportional to the number of variables
- In nearest neighbors, variance is inversely proportional to the number of neighbors

# Features and Targets

- ***Features:*** Input variables, predictors, or independent variables
  - In the data frame to the right: Sepal length, sepal width, petal length, petal width.
- ***Target***: What we want to predict or associate with what we measure: AKA outputs, response, dependent variable
  - In the data frame to the right: target

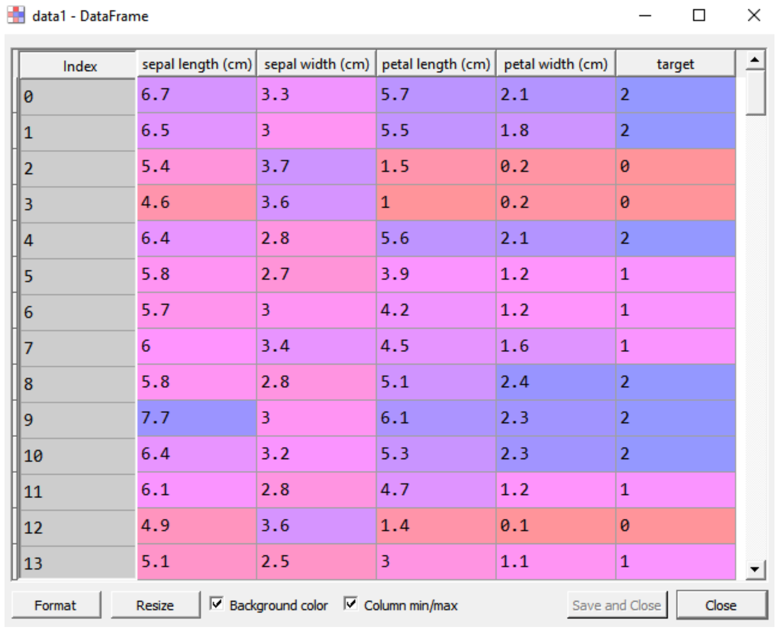


data1 - DataFrame

| Index | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|---|---|---|---|---|---|
| 0 | 6.7 | 3.3 | 5.7 | 2.1 | 2 |
| 1 | 6.5 | 3 | 5.5 | 1.8 | 2 |
| 2 | 5.4 | 3.7 | 1.5 | 0.2 | 0 |
| 3 | 4.6 | 3.6 | 1 | 0.2 | 0 |
| 4 | 6.4 | 2.8 | 5.6 | 2.1 | 2 |
| 5 | 5.8 | 2.7 | 3.9 | 1.2 | 1 |
| 6 | 5.7 | 3 | 4.2 | 1.2 | 1 |
| 7 | 6 | 3.4 | 4.5 | 1.6 | 1 |
| 8 | 5.8 | 2.8 | 5.1 | 2.4 | 2 |
| 9 | 7.7 | 3 | 6.1 | 2.3 | 2 |
| 10 | 6.4 | 3.2 | 5.3 | 2.3 | 2 |
| 11 | 6.1 | 2.8 | 4.7 | 1.2 | 1 |
| 12 | 4.9 | 3.6 | 1.4 | 0.1 | 0 |
| 13 | 5.1 | 2.5 | 3 | 1.1 | 1 |

# Supervised Learning

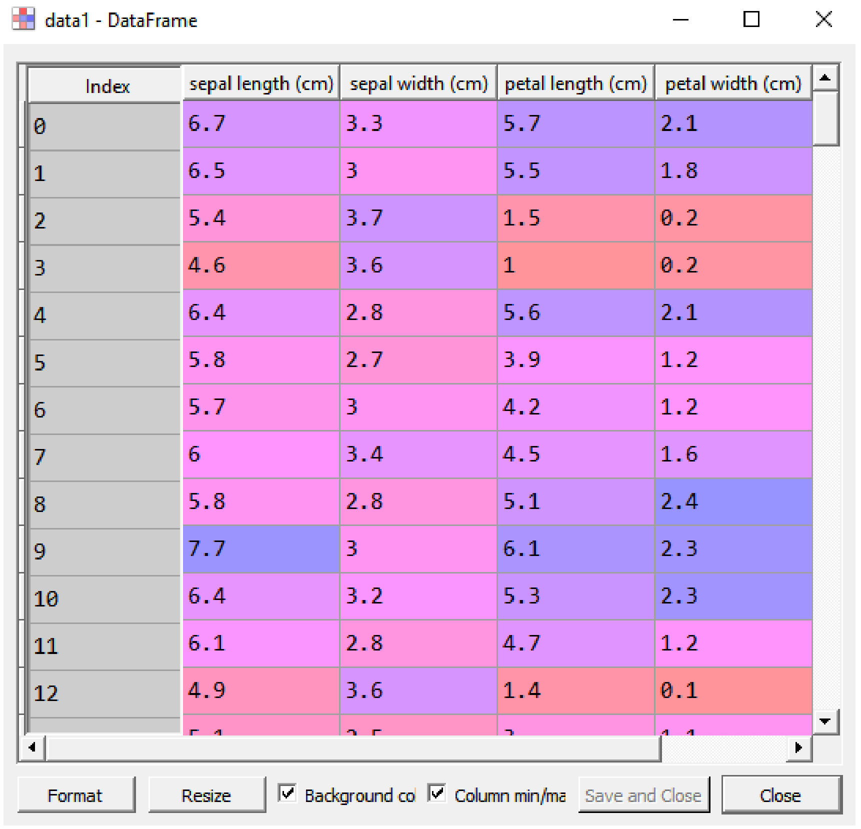
- *Supervised statistical learning* involves building a statistical model for predicting, or estimating, an output based on one or more inputs.
- Example supervised ML Algorithms:
  - Linear Regression
  - Decision Trees
  - Deep Learning
  - Support Vector Machine
- We will focus on supervised learning first


data1 - DataFrame

| Index | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|---|---|---|---|---|---|
| 0 | 6.7 | 3.3 | 5.7 | 2.1 | 2 |
| 1 | 6.5 | 3 | 5.5 | 1.8 | 2 |
| 2 | 5.4 | 3.7 | 1.5 | 0.2 | 0 |
| 3 | 4.6 | 3.6 | 1 | 0.2 | 0 |
| 4 | 6.4 | 2.8 | 5.6 | 2.1 | 2 |
| 5 | 5.8 | 2.7 | 3.9 | 1.2 | 1 |
| 6 | 5.7 | 3 | 4.2 | 1.2 | 1 |
| 7 | 6 | 3.4 | 4.5 | 1.6 | 1 |
| 8 | 5.8 | 2.8 | 5.1 | 2.4 | 2 |
| 9 | 7.7 | 3 | 6.1 | 2.3 | 2 |
| 10 | 6.4 | 3.2 | 5.3 | 2.3 | 2 |
| 11 | 6.1 | 2.8 | 4.7 | 1.2 | 1 |
| 12 | 4.9 | 3.6 | 1.4 | 0.1 | 0 |
| 13 | 5.1 | 2.5 | 3 | 1.1 | 1 |

# Unsupervised Learning

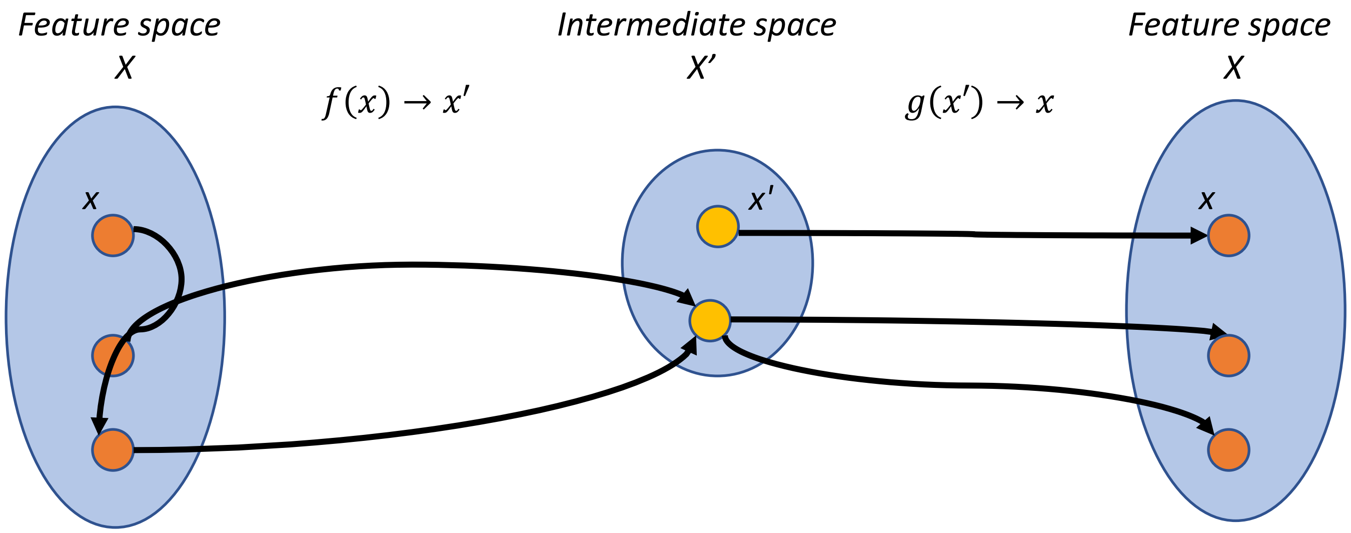
- With unsupervised statistical learning, there are inputs but no supervising output; nevertheless we can learn relationships and structure from such data.
- Example ML Algorithms
  - Hierarchical Clustering
  - K-Means
  - Mixture Models
  - Self Organizing Maps
  - Autoencoders



| Index | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| 0 | 6.7 | 3.3 | 5.7 | 2.1 |
| 1 | 6.5 | 3 | 5.5 | 1.8 |
| 2 | 5.4 | 3.7 | 1.5 | 0.2 |
| 3 | 4.6 | 3.6 | 1 | 0.2 |
| 4 | 6.4 | 2.8 | 5.6 | 2.1 |
| 5 | 5.8 | 2.7 | 3.9 | 1.2 |
| 6 | 5.7 | 3 | 4.2 | 1.2 |
| 7 | 6 | 3.4 | 4.5 | 1.6 |
| 8 | 5.8 | 2.8 | 5.1 | 2.4 |
| 9 | 7.7 | 3 | 6.1 | 2.3 |
| 10 | 6.4 | 3.2 | 5.3 | 2.3 |
| 11 | 6.1 | 2.8 | 4.7 | 1.2 |
| 12 | 4.9 | 3.6 | 1.4 | 0.1 |

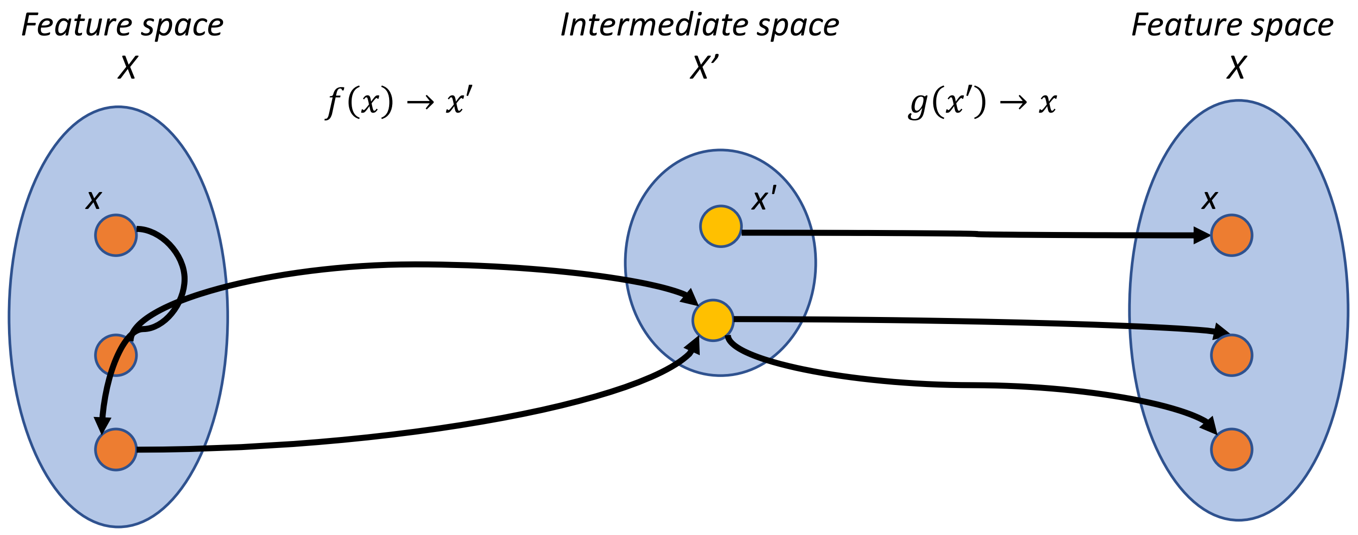
# Examples of unsupervised learning

- A learning a function that maps the input into an intermediate representation
- That intermediate representation makes the data *easier to interpret*
- There should be a map back from such intermediate representation and the original space
- The map from intermediate to feature space (*g*) should be as close as possible to the original input



# Examples of unsupervised learning

- Clustering: Cluster the diabetic patients into groups
- Example: K-Means
- Topic modeling: Describe the content of a set of documents (e.g., tweets) based on *topics* (soft clustering)
- Dimensionality reduction: Transform a large set of features into a smaller set of features while retaining the variability of the data
- Example Dimensionality Reduction: Principal Component Analyses (PCA)



## Supervised vs. Unsupervised Learning

- Supervised (See terminology in previous slides):
  - Each *X (observation)* is associated with a *Y (target)*
- Unsupervised (See terminology in previous slides):
  - We have *X* but no Y (target)
- Supervised learning is in general *easier* because we know how well we are building the association
- Unsupervised learning is harder because there is no clear evaluation method.
- This course will mostly deal with supervised learning.

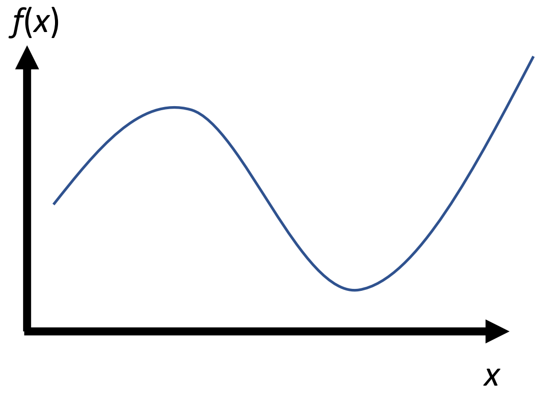
# Examples: Features and Targets

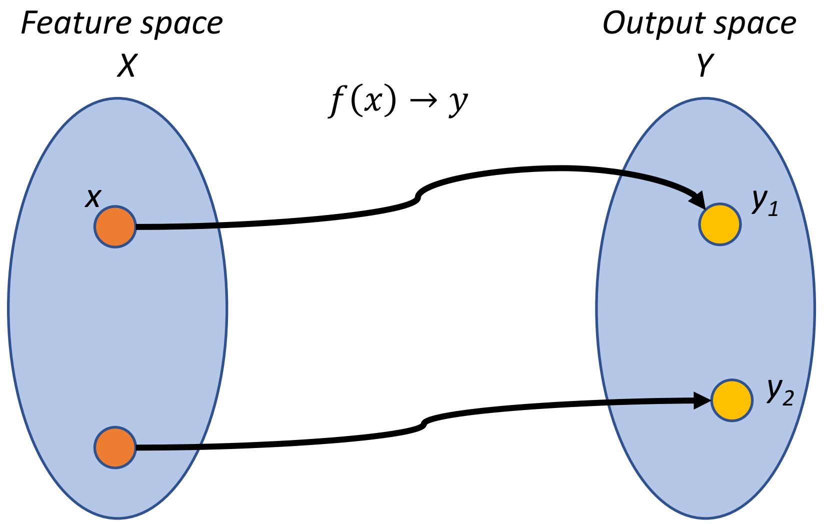
- Identify the features and targets in the following examples:
  - Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.
  - Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.
  - Identify the numbers in a handwritten ZIP code, from a digitized image.
  - Predict the longitude and latitude of a car based on GPS measurements, accelerometer, and gyroscope
  - Predict the hash tags of a tweet based on its text

# Mathematical formalization of learning

- We will say that **we want to learn a function *f* about a phenomenon**

A function takes one element ***x*** that belongs to the feature space ***X*** and maps it into *one and only one* element ***y*** in the output space ***Y***

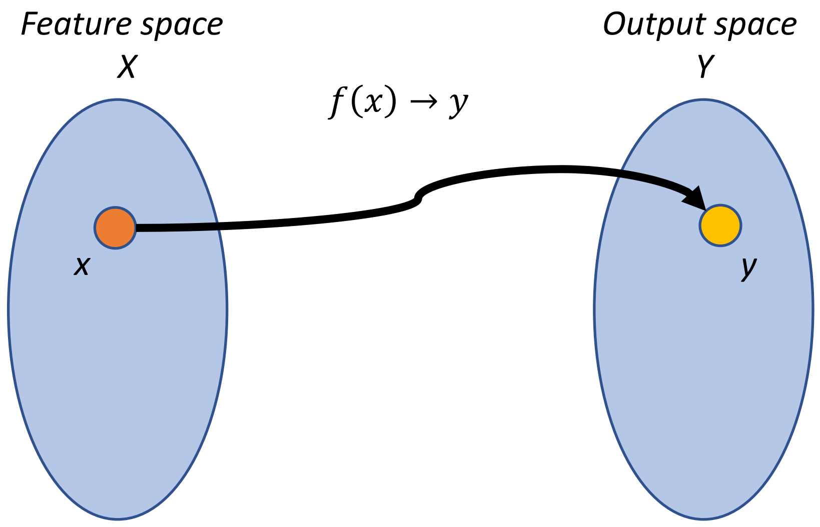




# Mathematical formalization of learning

- Predict the price of a stock in 6 months from now, based on company performance measures and economic data.

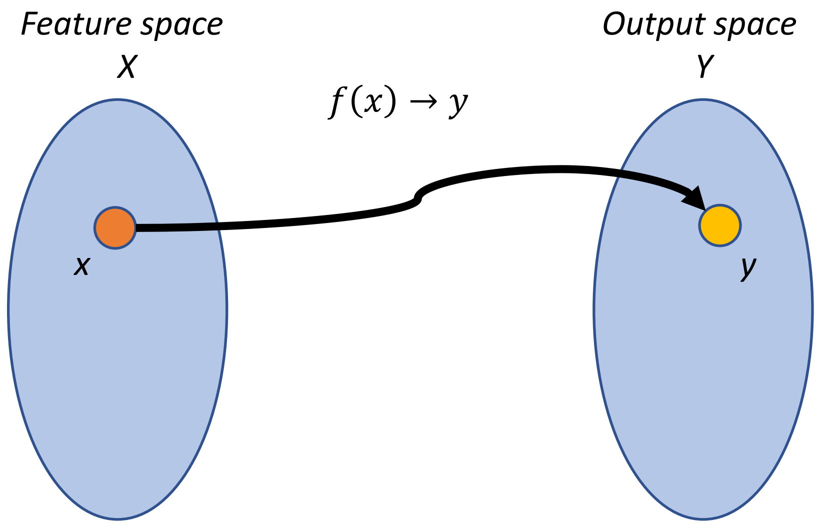
| | |
|---|---|
| *x*: | ? |
| *y*: | ? |
| *X*: | ? |
| *Y*: | ? |



## Mathematical formalization of learning

- Predict the price of a stock in 6 months from now, based on company performance measures and economic data.

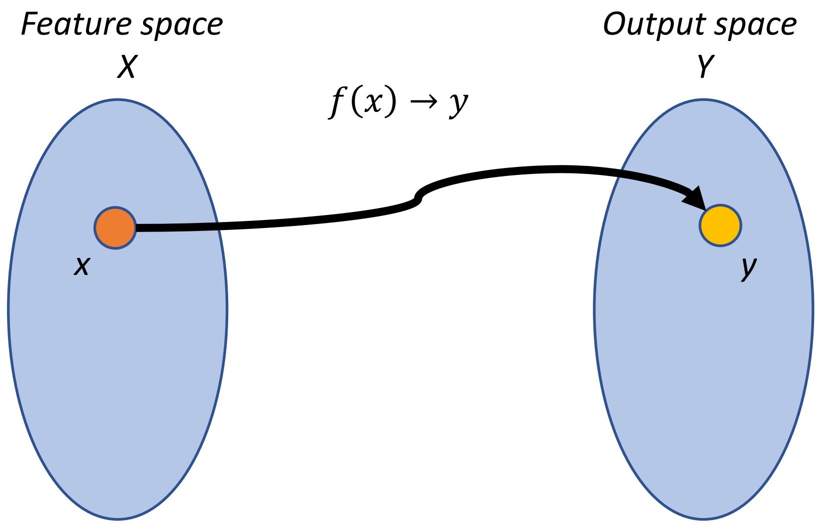
| | |
|---|---|
| *x*: | Market capitalization, trading volume, stock price, and economic data |
| *y*: | Specific stock price in 6 months |
| *X*: | Space of all possible market capitalization, volumes, stock prices, and economic data |
| *Y*: | Space of all possible stock prices in 6 months |



## Mathematical formalization of learning

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.

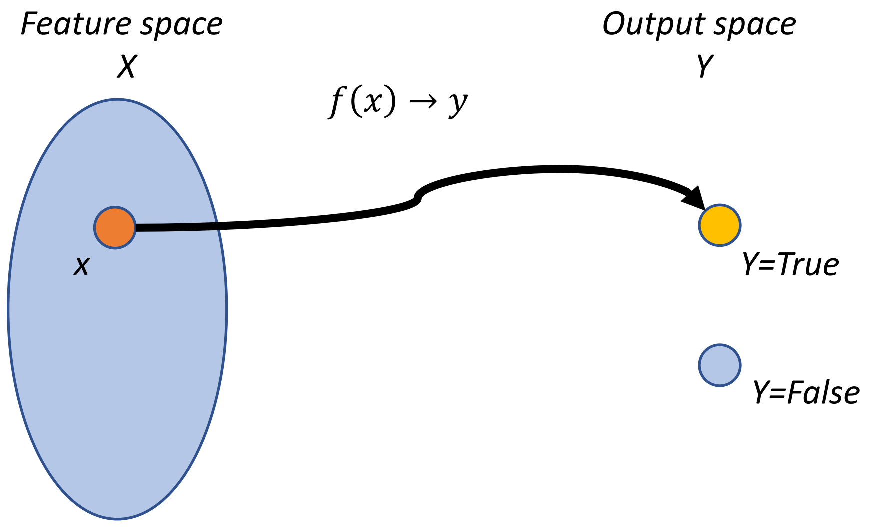
| | |
|---|---|
| *x*: | ? |
| *y*: | ? |
| *X*: | ? |
| *Y*: | ? |



# Mathematical formalization of learning

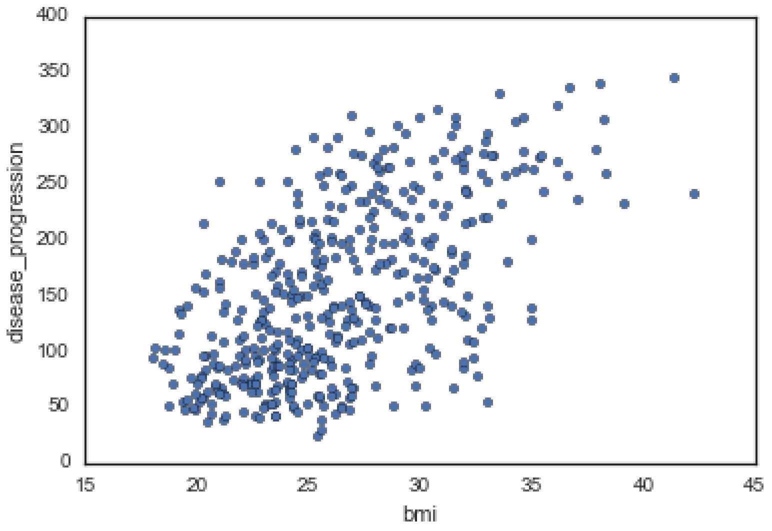
- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.

| | |
|---|---|
| *x*: | Specific instances of demographics, diet, clinical measure of a patient hospitalized due to a heart attack |
| *y*: | Whether specific patient will have a second heart attack |
| *X*: | Space of all instances of demographics, diet, and clinical data of patients hospitalized for heart attack |
| *Y*: | True or False |



# Example: Diabetes dataset

- Ten baseline variables including age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of *n* = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline
- What might we want to "learn" from this dataset?



# A statistical perspective on learning

- We wish to learn something from the data *D*
- Statistical learning means that the data have been generated by some unknown *random process*:

$$x, y \sim p(\,)$$

This reads x and y follow the probability distribution described by p.

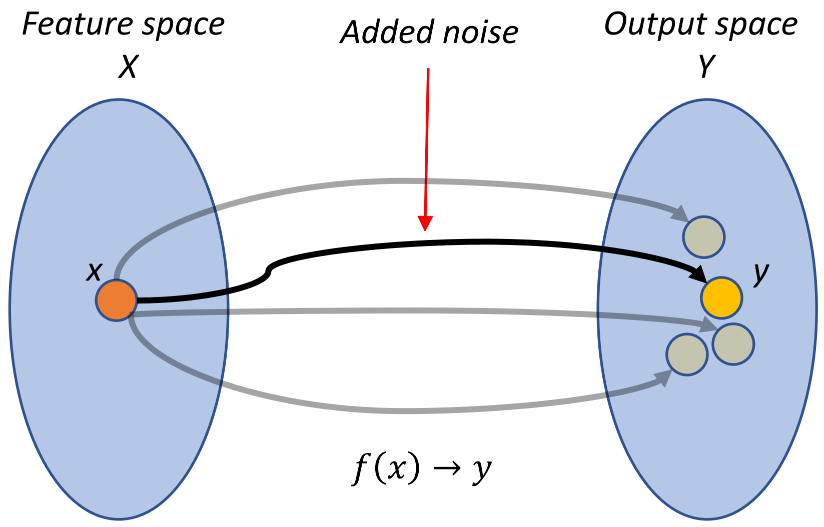
and **we wish to estimate the *unknown p* from such data**

# A statistical perspective on learning

- Statistical learning refers to a set of approaches for estimating *f* assuming that noise is added into the system and that the noise cannot be predicted from *x*

$$y = f(x) + \epsilon$$

$\epsilon$: error term (the noise)

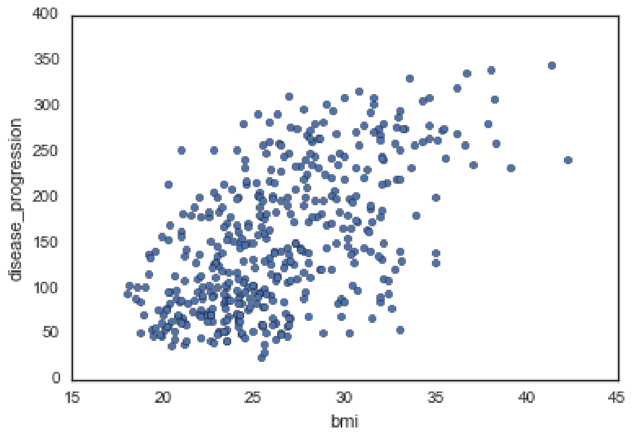


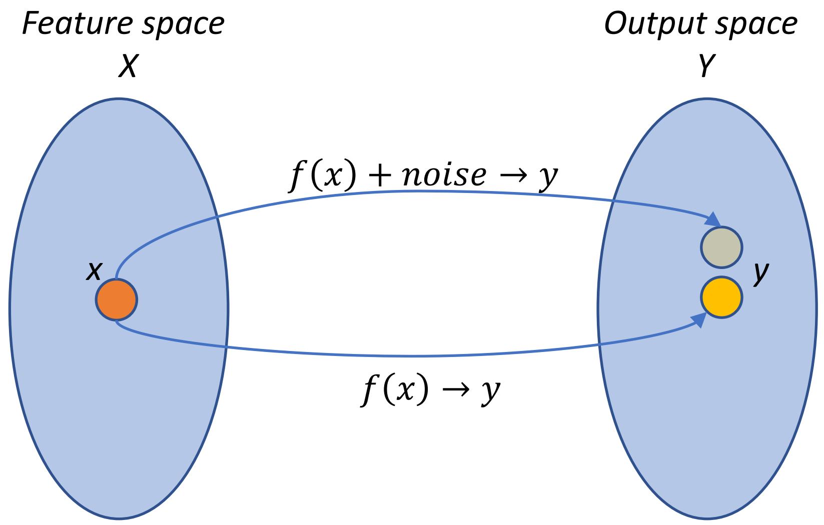
## Why we estimate *f*:

- Prediction
- Inference

## Why we estimate *f: Prediction*

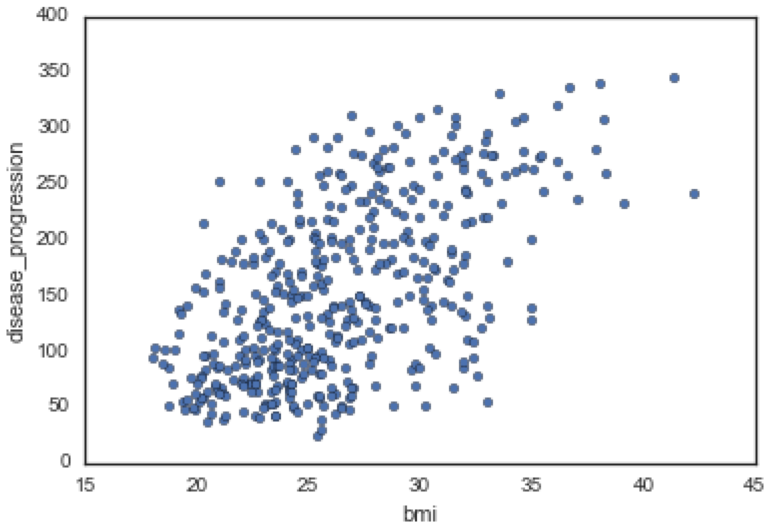
- **Prediction**: For a new data point never seen before: $\text{prediction} = \hat{Y} = \hat{f}(X)$
- E.g., for the diabetes dataset, we might want to predict disease progression based on the BMI of a patient.
- Noise is unpredictable and changes where we land in the output space randomly.





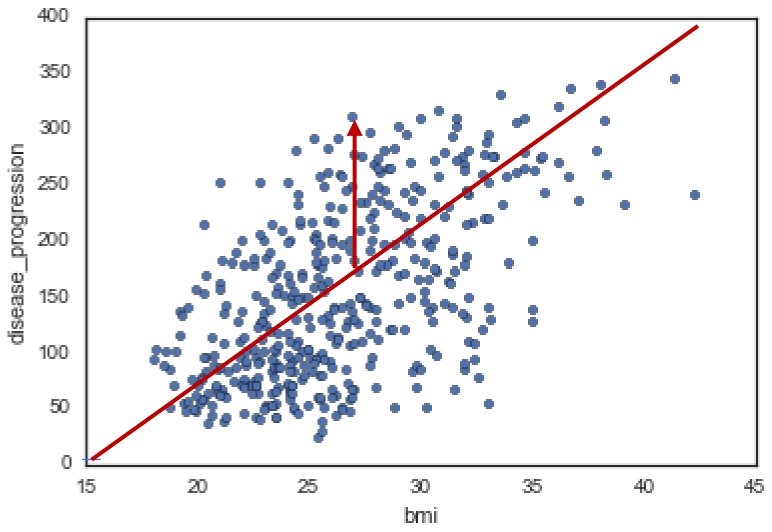
## Why we estimate *f*: Prediction

- Assuming that there is a function such that
$$y = f(x) + \epsilon$$
- We wish to minimize some loss function using our estimation
$$\hat{Y} = \hat{f}(X)$$
- How much can the loss function be reduced?



## Squared Error

- The diagonal line represents the best estimate of the cloud of data points.
- The best estimate is the line that minimizes the sum of the errors.
- One error is shown as the vertical line.
- The error is the distance between the diagonal line and the real data point.

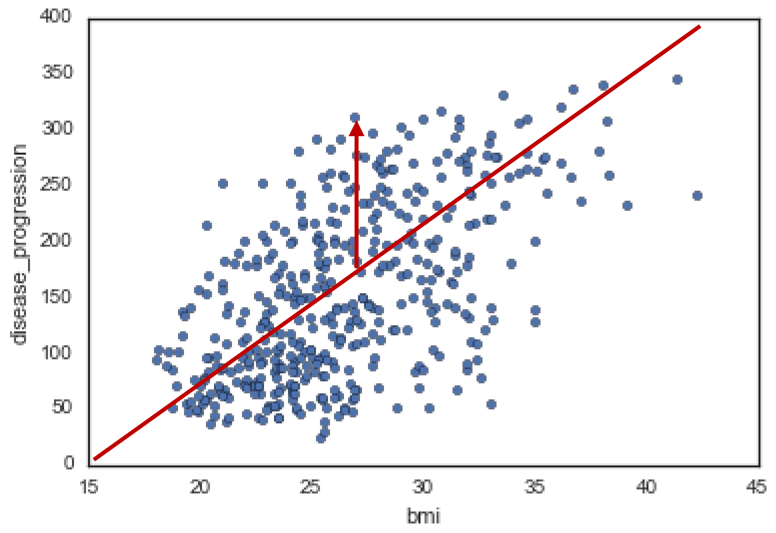


# Least Square Error Loss Function (LSE)

- Starting with simple linear regression: $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$
- $LSE = \sum_i (\hat{y}_i - \mathrm{y}_i)^2$
- Using calculus, we can show that $\hat{\beta}_0$ and $\hat{\beta}_1$ are minimized as follows:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$



## Expected Value of Squared Error

- Expected value of squared error yields individual reducible and irreducible error variances

$$E\left[\left(Y-\hat{Y}\right)^2\right] = E\left[\left(f(X)+\epsilon-\hat{f}(X)\right)^2\right]$$

$$= E\left[\left((f(X)-\hat{f}(X))+\epsilon\right)^2\right]$$, regroup

$$= E\left[\left(f(X)-\hat{f}(X)\right)^2 + 2\epsilon\left(f(X)-\hat{f}(X)\right)+\epsilon^2\right]$$, apply FOIL

# Expected Value of Squared Error

$E\left[\left(f(X) - \hat{f}(X)\right)^2\right] + E\left[2\epsilon\left(f(X) - \hat{f}(X)\right)\right] + E[\epsilon^2]$*, distribute* $E$

$E[\epsilon]$ *goes to 0 because* $\epsilon$ *is expected to be Gaussian and have 0 mean*

$= E\left[\left(f(X) - \hat{f}(X)\right)^2\right] + 0 + E[\epsilon^2]$, because E($\epsilon$) = 0

$E\left[\left(f(X) - \hat{f}(X)\right)^2\right]$ is **reducible** variance = LSE

Identity: $var(\epsilon) = E(\epsilon^2) - (E(\epsilon))^2$

Therefore, $E[\epsilon^2]$ is **irreducible** variance because $E[\epsilon] = 0$

## What is Reducible and Irreducible Error

- Suppose that x1, x2, x3, …, xN are characteristics of a patient’s blood sample.
- Suppose that Y is a variable encoding the patient’s risk for severe reaction to a drug
- Our goal is to predict $\widehat{Y}$ using X

## What is Reducible Error

- Reducible error is the error that can be reduced by optimizing hyper parameters of f(x)
- For example, the general form of a linear regression equation is

  $\hat{y} = mx + b$

  Where: **m** is the slope, **b** is the y intercept, **x** is the training data, and $\hat{\boldsymbol{y}}$ is the predicted value.
- Least squares error loss function: $LSE = \sum_i (\hat{y}_i - \mathrm{y}_i)^2$
- We can minimize the least squares loss function by selecting model parameters m and b such that LSE is minimized.

## What is Irreducible Error?

- The irreducible error ‘epsilon’ in the diabetes data set example could be due to:
  - The patient’s general health on a given day
  - Small manufacturing variations in the drug
  - Small measurement errors in x1, x2, x3, …, xN

## Why we estimate *f: Inference*

- **Inference**: Sometimes we want to understand *f* (look inside *f*)
  - Which predictors are important in the process of predicting Y:  e.g., do we need to include *gender* in the prediction of blood pressure?
  - Relationship between *Y* and each *X:*  e.g., blood pressure is higher for older people?
  - Is the relationship between *Y* and *X* appropriately captured by the model? Is the linear relationship enough?

## How do we estimate *f*?

- We estimate using **training data**.
- For notation, we will assume that we have *n observations* where each observation is a row in the data set.
- $x_{ij}$ is a specific row / column value in the data set: ‘i’ is the row and j is the column.
- $y_i$ is the independent variable / target / class for a specific observation.
- Two main methods for estimating F
  - Parametric Methods
  - Non Parametric Methods

## How do we estimate *f*: Parametric Methods

- Makes an assumption about the shape of *f*
- For example, assume the form of *f* is a linear model
- Use a procedure to fit or train the model and reduce the loss function for the model

## How do we estimate *f:* Nonparametric Methods

- Nonparametric methods don’t make assumptions about *f*.
- Example: Non parametric means that the model does not impose a specific function like linear regression (which assumes that y = mx + b)
- Informally, they try to get as close as possible to the training data but not too close.
- In general, the more data, the better the fit, but the harder to interpret.
- Example Non Parametric Method:
  - Random Forest

## How we estimate $f$

- Sometimes, the probability distribution that generated the data is known or assumed to be known except for a couple of parameters describing it.
- For example, say our target data follows a random normal distribution: $N \sim (\mu, \sigma)$
- In this example, we can estimate $f$ by estimating $\mu$ and $\sigma$.

$$X, Y \sim p(\mu, \sigma)$$

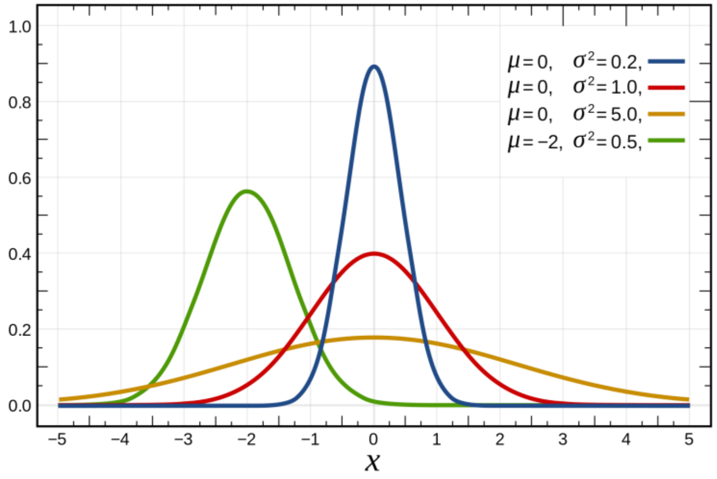
- We need to infer $\mu, \sigma$ from the training data

## Maximum Likelihood Estimation (MLE)

- Estimate the probability distribution parameters that are most likely to best represent the data.
- For example, if data follows the normal distribution, estimate the mean and standard deviation of the normal distribution that produces a distribution that is the closest match to the data.

# Gaussian distribution

- Gaussian or Normal distribution: most common probability distribution
- The Gaussian distribution is fully described by μ, $\sigma$



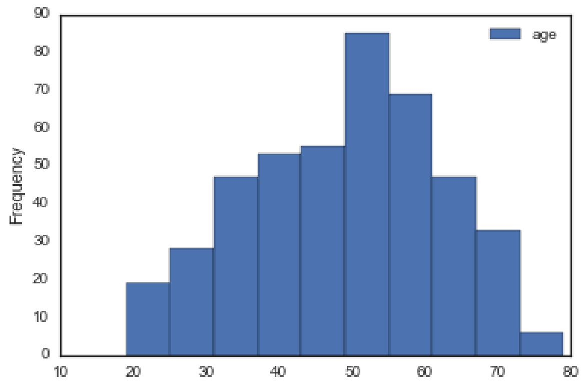
$$y \sim N(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2\sigma^2}(x-\mu)^2]$$

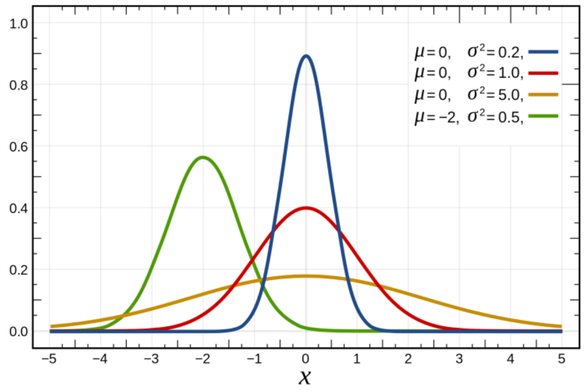
# Maximum Likelihood Estimation (MLE)

- Say $\hat{\theta}$ represents the mean and standard deviation of the normal distribution
- Say ‘y’ represents the data observations in a data set
- Then MLE of $\hat{\theta} = \arg\max p(y \mid \theta)$
- The MLE is maximized when the probability $p(y \mid \theta)$ is maximized
- The arg max function picks the maximum probability
- In the case of a Gaussian distributiuon, we are essentially using probability to find the $\mu, \sigma$ that provides the best fit to the data.

## A high-level description of MLE

- Given a data set that follows some probability distribution …
- We will use the Gaussian Normal distribution in this example but MLE can apply to any distribution
- Say we plot the data histogram and it looks Gaussian – like the data at the top right
- MLE is the process of using probability to find the μ, $\sigma$ that best fits the data (for a normal distribution)
- Essentially, MLE is the process of finding the general distribution (like on bottom right) that best fits the data at the top right.





## Derivation of Least Squares Using MLE

- The following slides on using MLE to derive the least squares equation uses information from the following stack exchange URL: https://stats.stackexchange.com/questions/253345/relationship-between-mle-and-least-squares-in-case-of-linear-regression

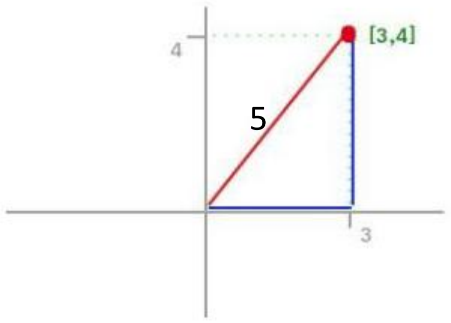
# Derivation of Least Squares Using MLE

- Y=X β + ϵ where $\epsilon \sim N(0, \sigma^2)$
- $Y \in \mathbb{R}^n, X \in \mathbb{R}^{nxp} and\ \beta \in \mathbb{R}^p$
- Note that the model error (residual) is $\epsilon = Y - X\beta$
- The goal is to find a vector of $\beta's$ that minimizes the L2 norm squared of the reducible error $\epsilon$.
- **Note: In this context, $\epsilon$ is the reducible error!**

$$\widehat{\beta}_{LS} = \underset{\beta}{\operatorname{argmin}} ||\epsilon||^2 = \underset{\beta}{\operatorname{argmin}} ||\mathbf{Y} - \mathbf{X}\beta||^2 = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i\beta)^2$$

## L2 Norm

- $L^2$ Norm calculates the shortest distance between the start
- Also known as the “Euclidian” Norm
- $\| x \|_2 = \sqrt{3^2 + 4^2} = 5$



## Derivation of Least Squares Using MLE

$$\widehat{\beta}_{LS} = \underset{\beta}{\operatorname{argmin}} ||\epsilon||^2 = \underset{\beta}{\operatorname{argmin}} ||\mathbf{Y} - \mathbf{X}\beta||^2 = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i\beta)^2$$

- Using the model derived on the previous slide (and shown above), we can derive the likelihood of the data given the parameters $\beta$ using the multiplication rule for independent probabilities.
- Assuming independent observations, that the joint probability distribution can be represented as the multiplication of the conditional probability for observing each data observation given the assumed distribution parameters (Gaussian in this example).

$$L(Y|X, \beta) = \prod_{i=1}^{n} f(y_i|x_i, \beta)$$

- Where $f(y_i|x_i, \beta)$ is the pdf of the normal distribution with 0 mean and variance $\sigma^2$
- $y_i$ = an individual outcome, $x_i$ = an individual observation, β = a regression coefficient.

## Derivation of Least Squares Using MLE

- Now plug the error equation $(y_i - x_i\beta)$ into the normal equation.
- And plug the normal equation into the likelihood equation:

$$L(Y|X,\beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - x_i\beta)^2}{2\sigma^2}}$$

## Derivation of Least Squares Using MLE

- When dealing with likelihoods, it’s generally easier to take the log because products become sums and exponentials go away.

$$L(Y|X,\beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - x_i\beta)^2}{2\sigma^2}}$$

- Applying the log transform to the equation on the previous slide (shown above) yields the equation below:

$$\log L(Y|X,\beta) = \sum_{i=1}^{n} \log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(y_i - x_i\beta)^2}{2\sigma^2}$$

## Derivation of Least Squares Using MLE

$$\log L(Y|X,\beta) = \sum_{i=1}^{n} \log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(y_i - x_i\beta)^2}{2\sigma^2}$$

- Since we want the maximum likelihood estimate, we want to find the maximum of the equation above with respect to $\beta$. The first term doesn't impact the estimate of $\beta$ so we can ignore it.

$$\widehat{\beta}_{MLE} = \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{n} -\frac{(y_i - x_i\beta)^2}{2\sigma^2}$$

$$\hat{\beta}_{MLE} = Maximum\ of\ MLE\ with\ respect\ to\ \beta$$

## Derivation of Least Squares Using MLE

$$\widehat{\beta}_{MLE} = \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{n} -\frac{(y_i - x_i \beta)^2}{2\sigma^2}$$

- Note that the denominator is a constant with respect to $\beta$ so we can ignore it. And, the maximum of a negative is like finding the minimum of a negative without the negative:

$$\widehat{\beta}_{MLE} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i \beta)^2 = \widehat{\beta}_{LS}$$

## Derivation of Least Squares Using MLE

- Recall that for this to work, we had to make certain model assumptions:
  - $\epsilon$ is normally distributed
  - $\epsilon$ has 0 mean and constant variance
- Under the above assumed conditions, MLE is equivalent to least squares.

## Derivation of Least Squares Using MLE

$$\widehat{\beta}_{MLE} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i\beta)^2 = \widehat{\beta}_{LS}$$

- Solving the MLE equation for $\beta$ and converting to matrix form (beyond the scope of this class) produces the equation that the computer uses to find the linear regression coefficients:

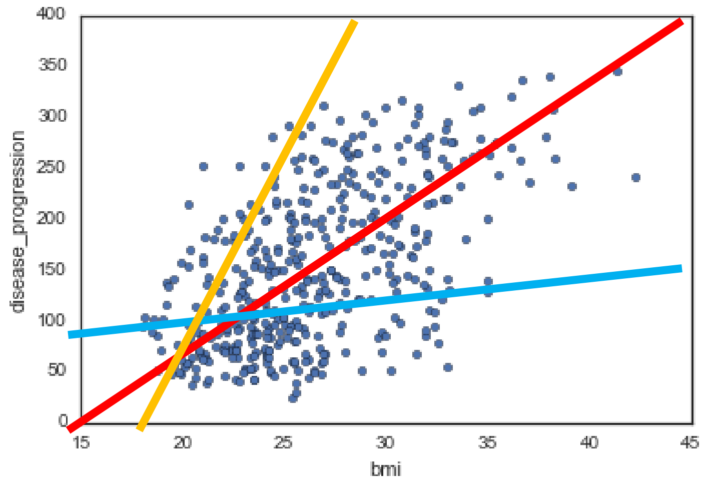
$$\beta = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

## Summary of MLE

- MLE is used to find the probability distribution parameters that make the distribution best represent the data
- If the distribution of the data is normal, we need to estimate the mean and standard deviation that makes the normal distribution best represent the data.
- The math to estimate the best parameters can be quite messy but the results can be quite simple.
- We typically do not do MLE calculations by hand; but rather, let the computer do the work for us.
- MLE can be applied to any probability distribution.

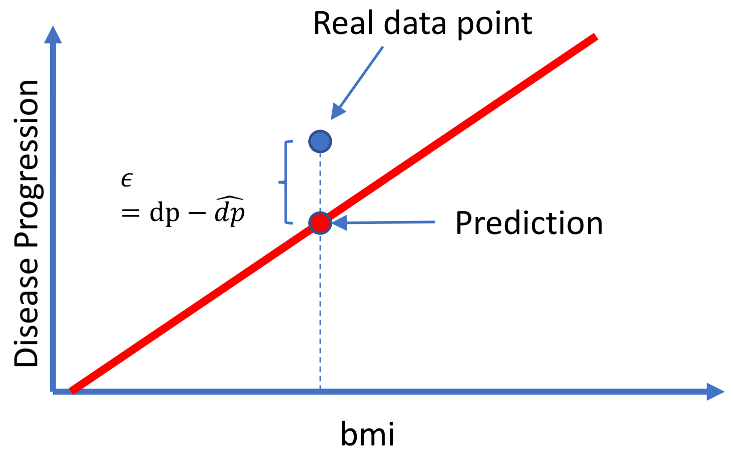
# Tying It All Together: Linear regression & Least Squares

- Assume we have a data set which has a body mass index predictor(X) and a diabetes disease progression target (Y).
- The goal is to find the line that is the best fit based on minimizing least squares.



# Tying It All Together: Linear Regression & Least Squares

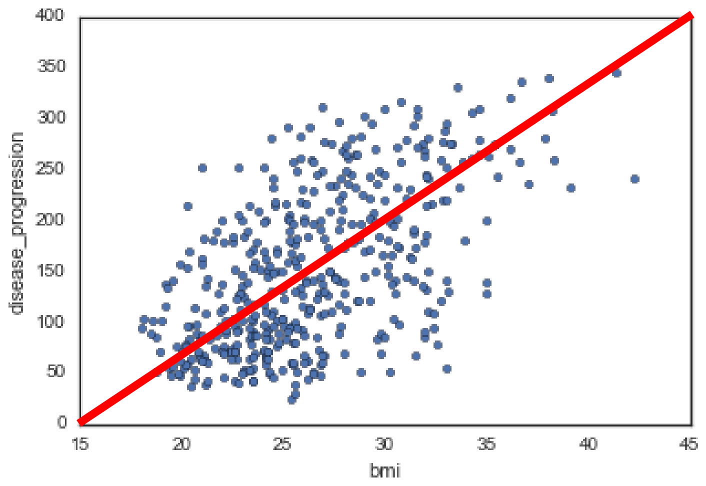
- The goal is to minimize summation of the squared error $\epsilon$ across all data points.
- The summation of the squared errors $\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \cdots + \epsilon_n^2$ is called the residual sum of squares or RSS.

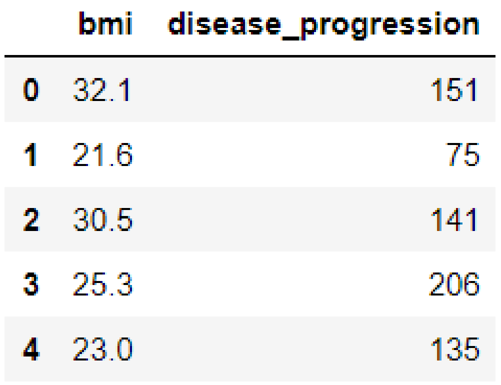


# Tying It All Together: Linear regression & Least Squares

- The computer will use the solution to the matrix form of the least squares equation to find $\beta_0$ and $\beta_1$ in the regression equation

$$\text{dp} = \beta_0 + \beta_1 * \text{bmi} + \epsilon$$



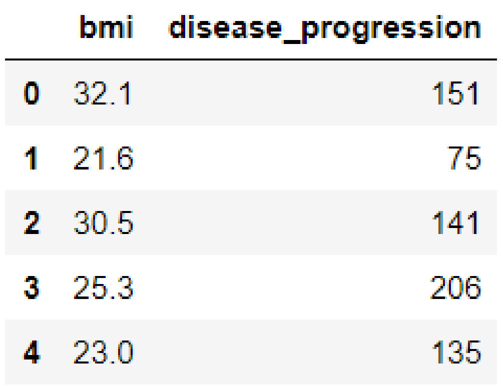


| | bmi | disease_progression |
|---|---|---|
| **0** | 32.1 | 151 |
| **1** | 21.6 | 75 |
| **2** | 30.5 | 141 |
| **3** | 25.3 | 206 |
| **4** | 23.0 | 135 |

## Tying It All Together: Linear regression & Least Squares

- Use the computer to solve the least squares equation for the vector of $\beta's$
- The $\beta's$ are the individual coefficients in the equation $Y = \beta X + b$
- The ‘bmi’ column is $X$
- The ‘disease_progression’ column is the $y$

$$\beta = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

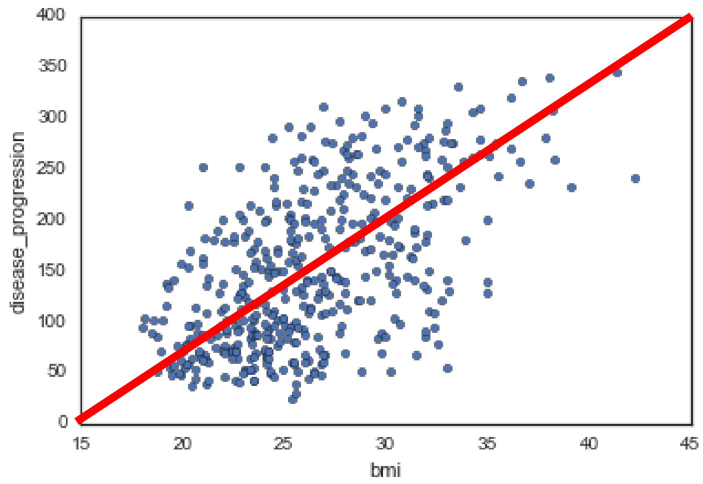


| | bmi | disease_progression |
|---|---|---|
| **0** | 32.1 | 151 |
| **1** | 21.6 | 75 |
| **2** | 30.5 | 141 |
| **3** | 25.3 | 206 |
| **4** | 23.0 | 135 |

# Tying It All Together: Linear regression & Least Squares

- The resulting equation can be used to make predictions
- Plug in a new ‘bmi’ to find a predicted disease progression

$$\mathrm{dp} = -117 + 10.23 * \mathrm{bmi} + \epsilon$$



## Generalization performance

- Generalization is the performance of a learning method on independent **test data**
- Generalization performance guides the choice of learning method or model
- **Why don't we teach the *best method*?**
    - David Wolpert: American Computer Scientist and Mathemetician
    - *Because there is no free lunch in statistics*: *David Wolpert*, "The Lack of A Priori Distinctions Between Learning Algorithms", 1996
    - A Priori: Reasoning or knowledge which proceeds from theoretical deduction rather than from observation or experience.

## Generalization performance (2)

- The no free lunch theorem implies that we need to
    1. Learn about the **particular dataset** we are working on (data science!)
    2. **Select the best method** using generalization performance (data science!)
- Essentially, there is no rule that maps the best model to each situation

## Measuring Generalization Performance: Theory

- We need to define a loss function $l(Y, \hat{f}(X))$
- Examples
  - Squared Error for regression
    $$l(Y, \hat{f}(X)) = \left(Y - \hat{f}(X)\right)^2$$
  - Zero-one loss for classification
  - $l\left(Y, \hat{f}(X)\right) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}\left(Y \neq \hat{f}(X)\right)$
    where I(*a*) is an "indicator" variable that equals 1 when the prediction is not equal to the true value (a == true).
  - If the indicator == 0, the *i*th observation was classified correctly.

# Measuring generalization performance: Test error and expected prediction error

- Test error is the prediction error over an *independent* test sample
$$\mathrm{Err}_T = \mathrm{E}\left[\, l(Y, \hat{f}(X)) \mid T \,\right]$$
where both *Y* and *X* are randomly sampled with a fixed training set *T*

- A related quantity is the expected prediction error
$$\mathrm{Err} = \mathrm{E}\left[l\left(Y, \hat{f}(X)\right)\right] = \mathrm{E}[\mathrm{Err}_T]$$
where everything is random including the training dataset.

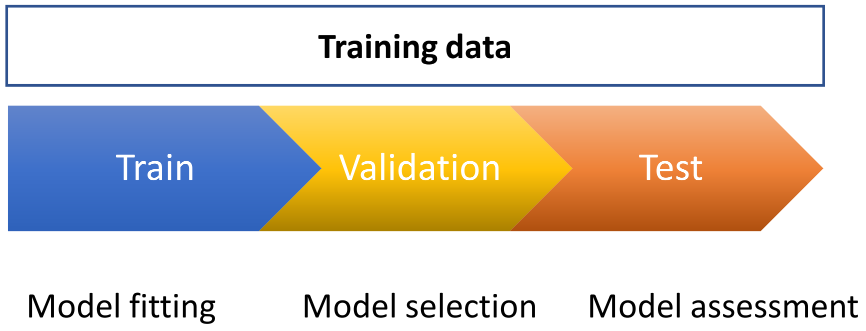
- Most methods effectively estimate $\mathrm{Err}$ instead of $\mathrm{Err}_T$.

# Estimating the expected test error

**Model select and assessment**

- Often, models have differing degrees of complexity controlled by a parameter $\alpha$ (e.g., $\hat{f}_\alpha(X)$)
- *Training data* is used to **fit** one model.
- *Validation* is used to select model hyper parameters and determines model **complexity**
- *Test* is used to estimate **expected test error**.

**Splits to estimate the expected test error**



## Estimating the expected test error: training data splits

- The **test split** should **only** be used at the end of the data science.
- If we compare two models in diabetes dataset: linear regression using BMI (M1) and linear regression using BMI and age (M2) (from the regression_model_comparison.ipynb)

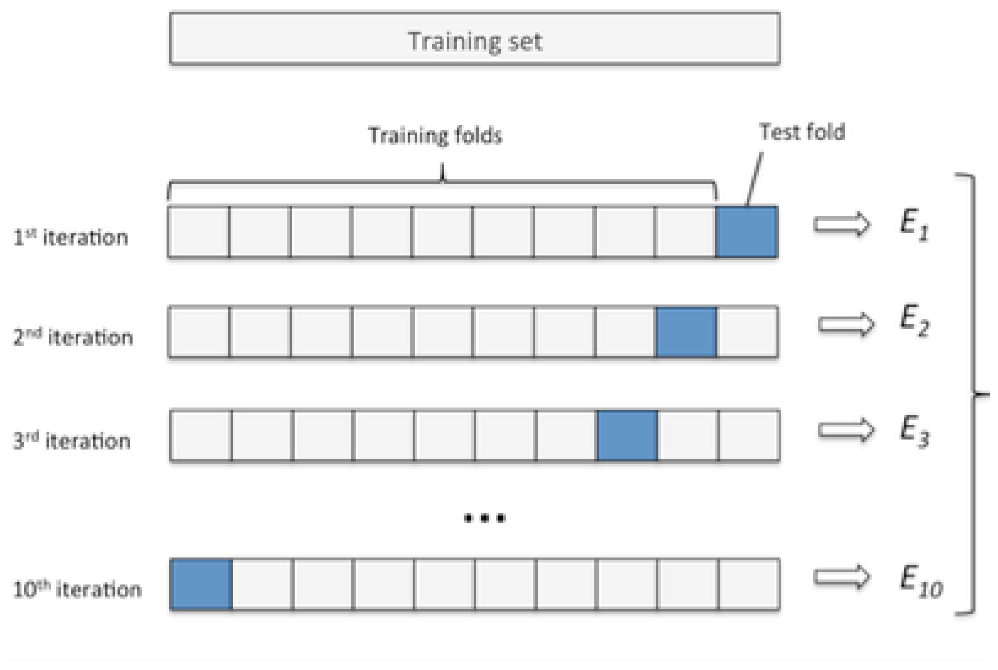
1. Model fit:
   M1 MSE (training) = 3723, **M2 MSE  (training) = 3680**
2. Model selection:
   **M1 MSE (validation) = 4582**, M2 MSE (validation) = 4618
3. Model assessment:
   **M1 MSE (test) = 3925**

# Estimating the expected test error: variable selection

- This procedure can be generalized to answer the question:
  - Which variables should be included in the model?
- One of the simplest such procedures is the forward stepwise selection algorithm (See ISLR 6.1.2):
  1. Start with no predicors
  2. For predictor_count from 0 to *num_predictors-1*
     1. Fit *num_predictors – predictor_count* models which add one variable to the current model
     2. If none of the models has lower validation error than current model, then break
     3. Add the predictor with the lowest validation error to the model
     4. Repeat until the next predictor in ‘1’ above does not improve model performance
  3. Return current model
- This procedure selects age, bmi, map, tc, ltg, and glu as the most important variable with estimated test error of **3324**

## Estimating the expected test error: n-fold cross validation

- The problem with the previous procedure is that we “don’t use” the validation and test splits to determine validation error (though we could do that).
- k-fold cross validation (partially) fixes this by running cross validation multiple times.



## Measuring generalization performance: classification

- It is typically *accuracy*

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(y_i, \hat{f}(x_i))$$

  where I(*a*, *b*) is 1 if *a* is equal to *b*, and 0 otherwise.
- What might be the problem with using accuracy?
  (hint: what is the accuracy of an algorithm that predicts that every email received is not a spam?)

## Confusion matrix

- Accuracy is misleading for *unbalanced* datasets
- **Activity**: Given one normal email and one spam email, GMAIL can perform a non-spam classification (NS) and spam classification (S). Where would you put all these cases in a **confusion matrix**?
- Assume positive = spam, and negative = not spam

| | | Predicted condition | |
|---|---|---|---|
| | | Condition positive (spam) | Condition negative (not spam) |
| True condition | Condition positive (spam) | S/S | S/NS |
| | Condition negative (not spam) | NS/S | NS/NS |

## Confusion Matrix

- Assume positive = not spam, and negative = not spam

|  |  | Predicted condition |  |
|---|---|---|---|
|  |  | Condition positive (spam) | Condition negative (not spam) |
| True condition | Condition positive (Spam) | True positive (TP) | False negative (FN) |
|  | Condition negative (not spam) | False positive (FP) | True negative (TN) |

## Common statistics from Confusion Matrix

- True positive rate (TPR), **Recall** or **Sensitivity**: TP / true condition positive
- True negative rate (TNR), **Specificity**: TN / true condition negative
- **Precision**: TP / predicted condition positive
- **Accuracy**: (TP + TN) / total population
- False Positive Rate (FPR): 1 – (TN / true condition negative) = 1 - TNR
- Prevalence: true condition positive / total population
- https://en.wikipedia.org/wiki/Confusion_matrix

| | | Predicted condition | |
|---|---|---|---|
| | | Condition positive | Condition negative |
| True condition | Condition positive | True positive (TP) | False negative (FN) |
| | Condition negative | False positive (FP) | True negative (TN) |

## Performance Measurement Interpretation

Assume a machine learning test was developed to predict diabetes.

- A group of study subjects is collected where some subjects are known to have diabetes and some do not have diabetes.
- A true prediction means that a subject is predicted to have diabetes
- A false prediction means that the subject is predicted to not have diabetes

# Performance Measurement Interpretation

Confusion matrix metric interpretation

- **TPR / Recall / Sensitivity**: Use when false positives are better than false negatives. Using this metric, it's better to label a healthy subject diabetic rather than a diabetic subject healthy.
- **TNR / Specificity**: Use when you want to minimize false positives. Using this metric, you don't want to classify any subjects with diabetes that do not have diabetes.
- **Precision**: Use in cases where high confidence in true positives is desirable. Maximizes the chance that a positive diabetes prediction is correct at the expense of missing some subjects that actually have diabetes.
- **Accuracy**: Only works well for balanced data sets. Probably not appropriate for a diabetes test.

## Receiver Operating Characteristic (ROC) Curve

- ROC was originally developed during world war 2 to measure the ability of radar receiver operators to distinguish between real aircraft targets and “false alarms” on the radar screens.
- ROC is typically used for models that predict the probability of 2 class outcomes
- **The key to understanding ROC**
  - **TPR and FPR values are calculated across a range of thresholds where the threshold is compared against the model probability prediction to determine the class**
  - **The resulting data is plotted as TPR vs. FPR**

# Receiver Operating Characteristic (ROC) Curve

- The ROC curve uses TP and FP cells in the confusion matrix (shown in red).
- A threshold is used to determine spam vs. not spam (see threshold col in table below)
- The resulting plot is called a ROC curve

| Email | P(spam) | Threshold<br>Spam if P(spam) > 0.5 |
|---|---|---|
| Normal | 0.1 | F |
| Spam | 0.4 | F |
| Normal | 0.2 | F |
| Spam | 0.6 | T |
| Normal | 0.7 | T |

| | | Predicted condition | |
|---|---|---|---|
| | | Condition positive (spam) | Condition negative (normal) |
| True condition | Condition positive (spam) | **TP** | FN |
| | Condition negative (normal) | **FP** | TN |

## ROC Curve

We will classify as spam if classifier predicts with more than 50% spam

- TPR: TP / true condition positive
- FPR: 1 – (TN / true condition negative) = 1 - TNR
- Populate the confusion matrix from the prediction results

| True Condition | P(spam) | Predicted Condition<br>Spam if P(spam) > 0.5 |
|---|---|---|
| Normal | 0.1 | Normal |
| Spam | 0.4 | Normal |
| Normal | 0.2 | Normal |
| Spam | 0.6 | Spam |
| Normal | 0.7 | Spam |

TPR = 1 / 2
FPR = 1 – (2 / 3)

| True condition | | Predicted condition: Condition positive (spam) | Predicted condition: Condition negative (normal) |
|---|---|---|---|
| | Condition positive (spam) | 1 | 1 |
| | Condition negative (normal) | 1 | 2 |

## ROC Curve

We will classify as spam if classifier predicts with more than 10% spam

- TPR: TP / true condition positive
- FPR: 1 – (TN / true condition negative) = 1 - TNR
- Populate the confusion matrix from the prediction results

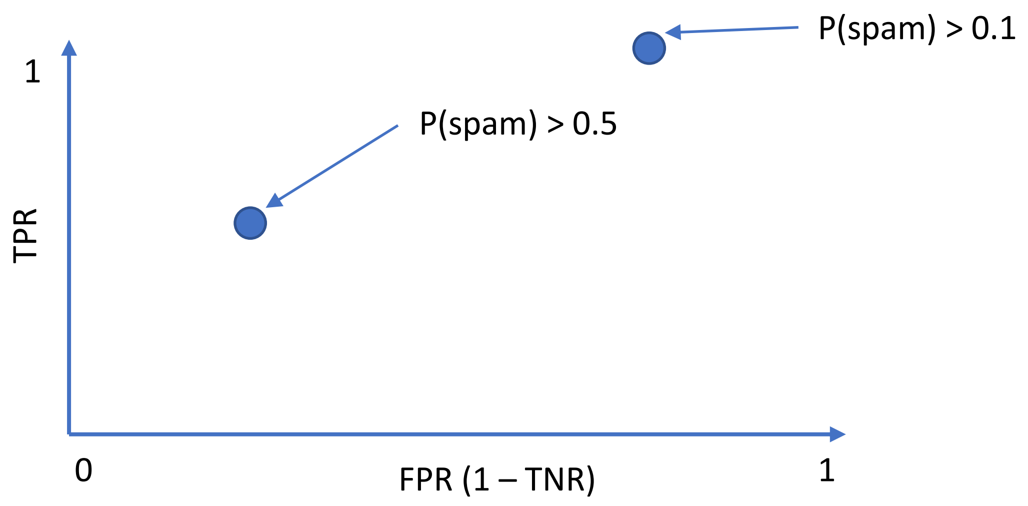
| True Condition | P(spam) | Predicted Condition Spam if P(spam) > 0.1 |
|---|---|---|
| Normal | 0.1 | Normal |
| Spam | 0.4 | Spam |
| Normal | 0.2 | Spam |
| Spam | 0.6 | Spam |
| Normal | 0.7 | Spam |

TPR = 1
FPR = 1 – (1 / 3)

| | | Predicted condition | |
|---|---|---|---|
| | | Condition positive (spam) | Condition negative (normal) |
| True condition | Condition positive (spam) | 2 | 0 |
| | Condition negative (normal) | 2 | 1 |

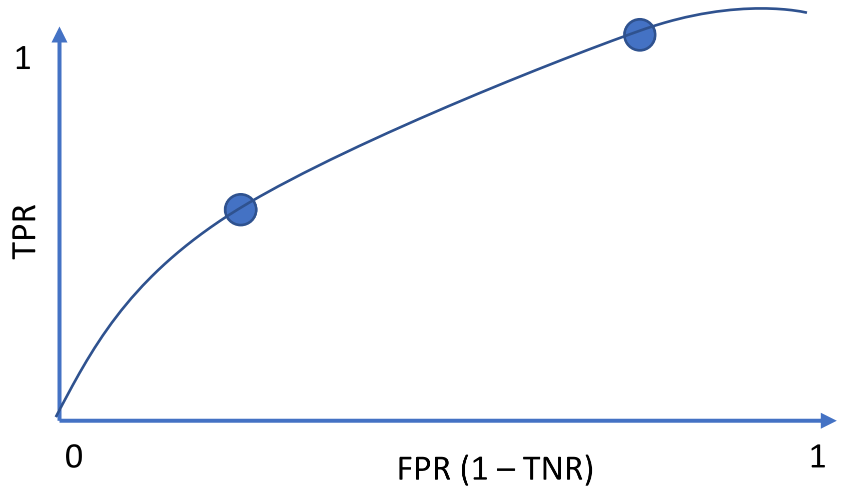
# Measuring generalization performance: ROC curve

- In the previous 2 slides, we calculated the TPR and FPR for thresholds 0.1 and 0.5
- In practice, we would calculate TPR and FPF for many more thresholds than just 2
- In this example, we show the two thresholds in the plot below



# Measuring generalization performance: ROC curve

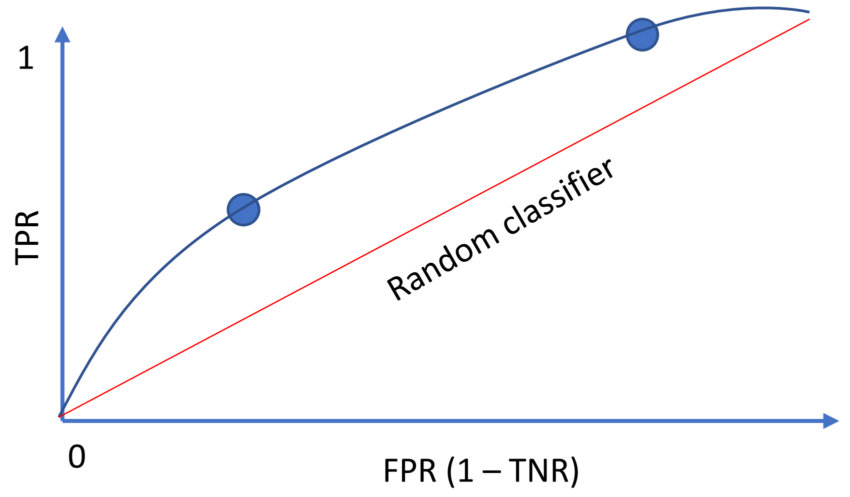
- If we connect a smooth line between the 2 points in this example, we can visualize what the ROC curve might look like if we calculated TPR and FPR for more than just 2 thresholds.
- Using the ROC Curve, we can pick threshold values that optimize a desired outcome.
- Example: It might be better for a medical test to allow more false positives so the test doesn't miss a true positive



Activity: Which point in the plot above is better for the medical test?

# Measuring generalization performance: ROC curve (2)

- Plot TPR vs. FPF for every threshold value
- The Area Under the Curve (AUC) represents the performance
- Higher AUC means better performance
- The AUC for the red diagonal line represents the "random guess" performance.



1

## Classification in diabetes example

- Logistic regression model:

$$p(\, y \mid X \,) = \theta^{y}(1-\theta)^{1-y}$$

where $\theta = \mathrm{sigmoid}(\mathrm{X}\beta)$

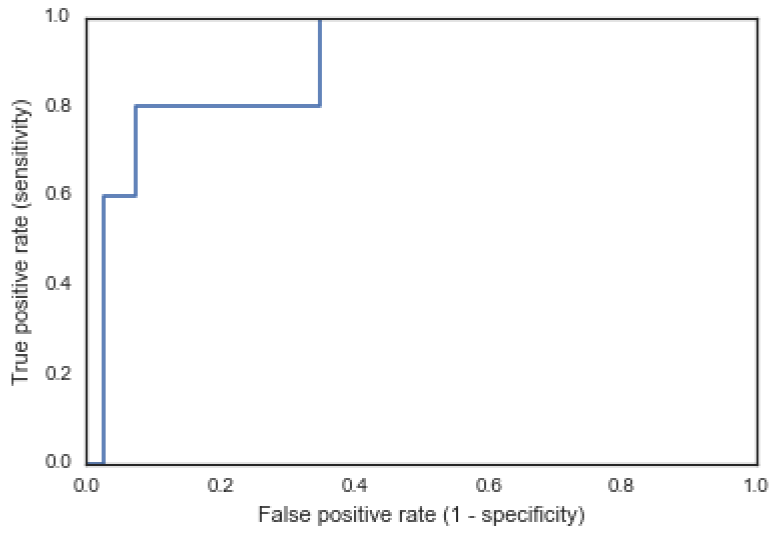
- And

$$\mathrm{sigmoid}(z) = \frac{1}{1+\exp(-z)}$$

- Finding the best $\beta$ is also done by maximum likelihood estimation!

## Classification in diabetes example

- Let’s assume that disease progression greater than 250 is really bad
- We want to predict really bad progressions
- Unbalanced because 10% of people have bad levels of progression



Area under the curve = 0.9
Using all predictor variables

## The Bayesian classifier

- Many values from confusion matrix are misleading because we are not using *prevalence rates.*
- Example: Unbalanced data.  Measuring prevalence of felons in a prison, measuring prevalence of sick people in a hospital, measuring prevalence of bad loans in a loan pool.
- Maximum likelihood estimators (MLE) are a “Frequentist” approach
- MLE estimates the most likely classification of the data given a hypothesis (bad disease progression or not)
- MLE is the estimate of $p(\text{ data } | \text{ hypothesis })$

## The Bayesian classifier

- From previous slide: MLE is the estimate of $p(\text{ data } | \text{ hypothesis })$
- What we really want is the **probability of the disease progression given the data.** This is called a Bayes classifier (see ISLR eq 2.10)

$$p(\text{ hypothesis } | \text{ data }) \propto p(\text{ data } | \text{ hypothesis })p(\text{hypothesis})$$

- The Bayesian classifier has the lowest possible error rate, but assumes that we know the population prevalence rate – which is often not available.

## The Bayesian classifier

- Sample made up data:
- Pretend we fit a logistic regression model to predict bad disease progression.
- Assume a threshold of 50% produces the following performance:
  - TPR / Sensitivity: 60%
  - TNR / Specificity: 95%
- This is very misleading if the prevalence of bad disease is only 10% in an unbalanced data set.
- For example, if someone is predicted to have a bad disease progression, then what does Bayes predict?

$$p(\text{ bad progression } | \text{ predicted bad disease }) = 57\%$$

Which is really different from 95% specificity!

## References

**An Introduction to Statistical Learning** - Gareth James • Daniela Witten • Trevor Hastie Robert Tibshirani